Conteinerização com Kubernets

Ielom Xavier David, Luker Almeida, Luiz Antônio, Antônio Frederico, Miguel Lopes, Jonas Grossi.

**Criação de um cluster de containers Docker na plataforma Raspberry Pi orquestrados pelo Docker Swarm**

**Introdução**

Containers (ou contêineres), no contexto de computação, são ambientes portáveis e isolados que permitem aos desenvolvedores empacotar aplicações com todos os links e bibliotecas que as aplicações demandam. O uso desta tecnologia melhora a eficiência das equipes de software, porque facilita o trabalho conjunto de seus integrantes, ao mesmo tempo em que reduz despesas com comunicações.

O Docker é uma plataforma de containerização extremamente útil e bem popular atualmente. Não por isso, o mesmo é dito como o projeto mais popular . O mesmo é distribuído pela empresa homônima, em duas “versões”, a Community Edition e a Enterprise Edition. A primeira, de distribuição Open Source, é a mais comumente utilizada pelos desenvolvedores independentes, sendo esta a versão que utilizaremos em nossa implantação.

Muitas vezes é necessário gerenciar os containeres e todos os aspectos inerentes aos mesmos. Nesse caso, preferencialmente, utilizamos soluções baseadas em orquestradores. O Docker Swarm é um exemplo de orquestrador. O mesmo permite a construção de clusters baseados na plataforma Docker, sendo distribuído juntamente com a instalação padrão do Docker. Uma alternativa ao Swarm, também como orquestrador, seria o Kubernetes, porém, em nosso caso, manteremos utilizando o Swarm.

Já o Raspberry Pi (usaremos a abreviação RPi) é um computador portátil (de tamanho próximo à um cartão de crédito), de baixo custo e alta usabilidade, desenvolvido principalmente para motivar o ensino de programação/hardware. Diversos projetos em computação podem ser desenvolvidos através desta plataforma, por exemplo, um computador completo, uma rede de sensores, servidor e até mesmo clusters, este último, sendo nosso caso. Além do RPi, existem alternativas (algumas, mais atraentes em diversos sentidos, como financeiro, por exemplo) como o Orange Pi.

**Motivação**

Neste projeto, nossa motivação será a construção de um cluster Docker, sobre a plataforma RaspberryPi. Além disso, pretendemos demonstrar as capacidades de tais tecnologias e por fim, adquirir conhecimentos sobre como operar com as mesmas.

**Detalhes**

A respeito da parte mais “técnica” do que foi desenvolvido durante este trabalho, iremos realizar tal detalhamento a seguir.

* Quanto ao Software:
  + SIstema Operacional Raspbian, versão
  + Sistema Operacional PulpStone (OpenWRT), versão
  + Docker, versão
  + Docker Visualizer (Container)
  + Etcher/dd
  + SSH
* Quanto ao Hardware:
  + 1X Placa Raspberry Pi Zero
  + 1X Placa Raspberry Pi 3 B
  + 1X Placa Orange Pi Zero (como roteador)
  + Computador, para codificação e gerenciamento do projeto
  + 3X Cabo Usb
  + 3X Cartão de Memória 16GB
  + Switch USB

Vale ressaltar que, para fins de real aproveitamento do Swarm, terão de ser atribuídos ao cluster pelo menos duas máquinas (placas).

**Definições**

  Iniciamos a parte prática do Trabalho sobre kubernetes na data de 23/04/2018 onde criamos a máquina virtual usando o programa virtualbox para instalar o ubuntu mate 16.04  64 bits após isso iniciamos a instalação do docker (sem problemas através do docker conseguimos rodar node js para teste) e o kubernetes (tivemos um problema que vai ser explicado abaixo).

Na semana  do dia 30/04/2018 começamos o estudo do mini-kube e tentamos implantar na máquina virtual (onde surgiu se um problema devido a virtualização da máquina virtual, estamos usando uma máquina virtual para instalação do kubernetes, onde surgiu se um problema devido a necessidade de virtualização de uma nova máquina virtual dentro da mesma ).

Após o problema começamos a usar o orange pi para usar o mini kube

No dia 21/05/2018  estamos realizando a finalização da instalação do docker e kubernetes no Orange pi  modelo Orange pi PC plus(3) e Orange pi zero plus(1 usado como roteador) usando os seguintes comandos:

Comando de instalaçao do docker:

# Como usuário root, faça:

# docker -version

# docker

# curl -sSL https://get.docker.com | sh

# docker

# docker run hello-world

No dia 22/06/2018-Ao recriar o Docker foi necessário fazer a instalação do portainer.io

O portainer.io é tipo o Kubernetes, mas não dá nenhum problema pra instalar porque ele roda dentro do próprio Docker.

**Instalação**

Consideramos a seguinte infraestrutura para nosso projeto:

* Raspberry Pi Zero como Node Master
* Raspberry Pi 3 como Worker
* Orange PI Zero como roteador, no gerenciamento de rede do sistema
* Cada um dos cartões de memória será atribuído à uma das placas

Após instalados os sistemas (Raspbian, para os Raspberry e PulpStone para o OrangePI), podemos ligar os mesmos. Inicialmente, vamos configurar o Roteador PulpStone, de preferência, atribuindo IPs estáticos para os Raspberry. Também é interessante configurar os hostnames das placas com nomes mais “didáticos”. Ademais, para fins de teste, pode-se configurar o roteador conforme a preferência. Também será necessária uma conexão com a Internet, para podermos fazer o download de artefatos necessários.

Conecte-se à rede do roteador PulpStone por seu computador principal. Iremos utilizar, para operar sobre os Raspberry, uma conexão SSH. Podemos utilizar a conexão como usuário preconfigurado “pi”. Conecte-se primeiro no Raspberry Pi Zero, que será nosso Master.

Vamos instalar o Docker, e juntamente por padrão, o Docker Swarm. Faremos isto através destes comandos:

*curl -sSL https://get.docker.com | sh*

*sudo usermod -aG docker pi*

Habilite o serviço e reinicie o sistema:

*sudo systemctl enable docker*

*sudo reboot -h now*

Será necessária a reautenticação pelo SSH. Verifique a instalação pelo comando:

docker ps

Repita este processo para os demais Nodes do Cluster (em nosso caso, para cada placa).

Para fins de deixar nossa instalação visualmente mais atraente, utilizaremos o Visualizer, uma interface gráfica web que permite visualizar detalhes sobre nossos containers Docker. O mesmo está disponível pelo Docker Hub, repositório de containers Docker. Lembre-se que isto será necessário ser feito na placa Master. Faremos:

*docker pull alexellis2/visualizer-arm*

Feito isto, já é possível para nós iniciarmos nosso cluster Swarm. Faremos isto através do comando (atenção ao IP, pois deve corresponder ao seu caso), a ser executado em sua placa Master. Note que o endereço deve corresponder, nesse caso, ao endereço da sua própria placa que está realizando a função de Node Master:

*docker swarm init --advertise-addr 192.168.1.106*

Uma mensagem será retornada, contendo um token que usaremos para adicionar os demais Nodes ao cluster. Para testar nosso estado atual usamos:

*docker node ls*

Que deverá retornar um node e seus detalhes, nesse caso, o Master. Continuando, vamos agora fazer uso do Visualizer, que fizemos o pull anteriormente. Tal inicialização será feita através do Master. Para inicializarmos um service contendo o Visualizer, devemos rodar o comando a seguir, dando atenção à porta escolhida. (em nosso caso, a opção “--publish 6660:80/tcp” utiliza da porta 80 para a “comunicação interna” do cluster e a porta 6660 para a comunicação, ou acesso, externo).

*docker service create --name cluster --publish 6660:8080/tcp --constraint node.role==manager --mount type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock alexellis2/visualizer-arm:latest*

Para verificar se tudo está nos conformes, podemos acessar o endereço de acesso do Master, seguido da porta.

*192.168.1.106:6660*

Com o Docker já instalado nas demais placas e o Master já configurado, nos resta nesta etapa configurar as demais placas para operarem juntamente ao Master, como Workers em nosso cluster. Esta configuração fará uso do token gerado durante a inicialização do Master. Para isto, utilizaremos do seguinte comando, a ser rodado em cada máquina:

*docker swarm join --token seu\_token\_aqui 192.168.1.106:2377*

Certifique-se aqui que não há problemas em relação à sincronização de relógios nas máquinas.

E isto conclui a construção de nosso cluster. Ademais, a seguir apresentaremos detalhes referentes a utilização do mesmo.

**Uso**

//Definições,Motivação,Detalhes,Instalação,Uso

//<https://www.filipeflop.com/blog/docker-swarm-e-cluster-com-raspberry-pi/>

//<https://hub.docker.com/r/itzg/minecraft-server/>

**Referências**

**Links pesquisados:**

**ferramentas usadas no estudo de kubernetes**

[**https://github.com/luxas/kubernetes-on-arm**](https://github.com/luxas/kubernetes-on-arm)

[**https://www.docker.com/what-docker**](https://www.docker.com/what-docker)

[**https://docs.docker.com/get-started/#recap-and-cheat-sheet**](https://docs.docker.com/get-started/#recap-and-cheat-sheet)

[**http://www.allitebooks.com/?s=docker**](http://www.allitebooks.com/?s=docker)

[**https://linuxceara.com/o-que-%C3%A9-docker-5f2bd86d0643**](https://linuxceara.com/o-que-%C3%A9-docker-5f2bd86d0643)

[**https://platform9.com/blog/kubernetes-docker-swarm-compared/**](https://platform9.com/blog/kubernetes-docker-swarm-compared/)

[**https://www.youtube.com/playlist?list=PLf-O3X2-mxDkiUH0r\_BadgtELJ\_qyrFJ\_**](https://www.youtube.com/playlist?list=PLf-O3X2-mxDkiUH0r_BadgtELJ_qyrFJ_)

[**http://john.e-wilkes.com/papers/2016-Queue-Kubernetes.pdf**](http://john.e-wilkes.com/papers/2016-Queue-Kubernetes.pdf)

[**https://logz.io/blog/rise-kubernetes-2017/**](https://logz.io/blog/rise-kubernetes-2017/)

[**http://file.allitebooks.com/20151213/Using%20Docker.pdf**](http://file.allitebooks.com/20151213/Using%20Docker.pdf)

[**http://techfree.com.br/2016/04/gerenciando-multiplos-containers-docker/**](http://techfree.com.br/2016/04/gerenciando-multiplos-containers-docker/)

[**https://www.redhat.com/pt-br/topics/containers/whats-a-linux-container**](https://www.redhat.com/pt-br/topics/containers/whats-a-linux-container)

[**https://www.dynatrace.com/news/blog/2018-year-of-kubernetes-with-help-from-docker/**](https://www.dynatrace.com/news/blog/2018-year-of-kubernetes-with-help-from-docker/)

[**https://www.youtube.com/watch?v=e\_eXUw\_z7H4&feature=youtu.be**](https://www.youtube.com/watch?v=e_eXUw_z7H4&feature=youtu.be)

[**https://docs.google.com/document/d/1rf60cJZY8-ZU8wxELUVqG3Ea2auSsFPKIvzeIRCKnBE/edit**](https://docs.google.com/document/d/1rf60cJZY8-ZU8wxELUVqG3Ea2auSsFPKIvzeIRCKnBE/edit)

[**https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/#why-do-i-need-kubernetes-and-what-can-it-do**](https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/#why-do-i-need-kubernetes-and-what-can-it-do)

[**https://kubernetes.io/case-studies/**](https://kubernetes.io/case-studies/)

[**https://www.airpair.com/docker/posts/8-proven-real-world-ways-to-use-docker**](https://www.airpair.com/docker/posts/8-proven-real-world-ways-to-use-docker)

[**https://www.ctl.io/developers/blog/post/what-is-docker-and-when-to-use-it/**](https://www.ctl.io/developers/blog/post/what-is-docker-and-when-to-use-it/)

**Estudo para Implementação**

[**https://github.com/ulm0/k8s-arm**](https://github.com/ulm0/k8s-arm)

[**https://github.com/eon01/DockerCheatSheet**](https://github.com/eon01/DockerCheatSheet)

**instalação do kubernetes**

  Nenhuma instalação do kurbertes para rm funcionou no orange pi (partes desses erros se da por conta da arquitetura do processador)

easy install do k82 pro arm seguinte erro:

ele dá erro na hora de instalar alguma coisa do docker

eu acho q chama docker-ce(Docker-ce = Docker Community Edition. É o utilitário principal)

mas ele dá erro tb,ele fala q o docker devia tá rodando pra instalar(coisa q ta acontecendo pq o docker tá instalado) e só para a compilação

Link da compilação do docker para arm:

**https://github.com/docker/docker-ce/blob/master/components/engine/project/ARM.md(FUNCIONOU)**

**Tentativa de instalação do kubernetes pelo repositório oficial:**

<https://kubernetes.io/docs/tasks/tools/install-kubeadm/#installing-kubeadm-kubelet-and-kubectl>

ocorreu o seguinte erro terminando a última etapa da instalação:

erro na instalação do docker.io

comando para o retorno do dpkg --configure -a:

Setting up docker.io (1.13.1-0ubuntu1~16.04.2) ...

addgroup: The group `docker' already exists as a system group. Exiting.

Job for docker.service failed because the control process exited with error code. See "systemctl status docker.service" and "journalctl -xe" for details.

invoke-rc.d: initscript docker, action "start" failed.

● docker.service - Docker Application Container Engine

 Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)

 Active: failed (Result: exit-code) since Sat 2018-05-12 19:17:53 UTC; 832ms ago

   Docs: [https://docs.docker.com](https://l.facebook.com/l.php?u=https%3A%2F%2Fdocs.docker.com%2F&h=ATMIgFGOz-Zrqe2Zoj9olAqNVRLW9D-NHUMOwSGygLY1DhTLGVsZCJgCFElo3QeFgTbTKFg_SOcQWRfj5SLNcZM3KO7wL39pk0qaenluu6-DsaaeylZf)

Process: 30029 ExecStart=/usr/bin/dockerd -H fd:// $DOCKER\_OPTS (code=exited, status=1/FAILURE)

Main PID: 30029 (code=exited, status=1/FAILURE)

May 12 19:17:52 orangepipc systemd[1]: Starting Docker Application Container Engine...

May 12 19:17:52 orangepipc dockerd[30029]: time="2018-05-12T19:17:52.647866365Z" level=info msg="libcontainerd: new containerd process, pid: 30035"

May 12 19:17:53 orangepipc dockerd[30029]: time="2018-05-12T19:17:53.662193879Z" level=fatal msg="Your Linux kernel version 3.4.113-sun8i is not supported for running docker. Please upgrade your...0.0 or newer."

May 12 19:17:53 orangepipc systemd[1]: docker.service: Main process exited, code=exited, status=1/FAILURE

May 12 19:17:54 orangepipc systemd[1]: Failed to start Docker Application Container Engine.

May 12 19:17:54 orangepipc systemd[1]: docker.service: Unit entered failed state.

May 12 19:17:54 orangepipc systemd[1]: docker.service: Failed with result 'exit-code'.

Hint: Some lines were ellipsized, use -l to show in full.

dpkg: error processing package docker.io (--configure):

subprocess installed post-installation script returned error exit status 1

Errors were encountered while processing:

docker.io

Atualizamos o kernel para tentar solucionar o problema mas continua  na mesma.

links de referencia para o update:

https://diyprojects.io/armbian-update-kernel-test-versions-nightly/#.Wx72M9VKiM9

outro link para tentativa de instalação do kubernetes:

<https://kubernetes.io/docs/tasks/tools/install-kubeadm/>

ocorreu o seguinte erro :

[init] Using Kubernetes version: v1.10.2

[init] Using Authorization modes: [Node RBAC]

[preflight] Running pre-flight checks.

[WARNING SystemVerification]: docker version is greater than the most recently validated version. Docker version: 18.05.0-ce. Max validated version: 17.03

[WARNING FileExisting-crictl]: crictl not found in system path

Suggestion: go get [github.com/kubernetes-incubator/cri-tools/cmd/crictl](https://l.facebook.com/l.php?u=http%3A%2F%2Fgithub.com%2Fkubernetes-incubator%2Fcri-tools%2Fcmd%2Fcrictl&h=ATMIgFGOz-Zrqe2Zoj9olAqNVRLW9D-NHUMOwSGygLY1DhTLGVsZCJgCFElo3QeFgTbTKFg_SOcQWRfj5SLNcZM3KO7wL39pk0qaenluu6-DsaaeylZf)

[preflight] Some fatal errors occurred:

[ERROR Swap]: running with swap on is not supported. Please disable swap

[preflight] If you know what you are doing, you can make a check non-fatal with --ignore-preflight-errors=...

root@orangepipcplus:~#

**resumo: Não era suportado o swap do linux este precisava ser desabilitado.**

Desabilitamos  o swap logo surgiu o problema onde a versão do docker instalado não era compatível com o kubernetes logo instalamos versao 17 do dockers

usamos o seguinte link  para isso:

<https://stackoverflow.com/questions/47018164/how-to-install-an-specific-version-of-docker-on-amazon-linux-ami>

Descobrimos que kubernetes não funciona na arquitetura do orange pi  pois procuram alguns registradores que não são encontrados nessa arquitetura.

Descobrimos isso após tentarmos usar o kubernetes onde esse e divido em 3 programas

sendo que o principal não inicia se os outros dois não estiverem funcionando.

ocorrendo o seguinte erro:

Cgroup Driver: cgroupfs

[Service]

Environment="KUBELET\_KUBECONFIG\_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf"

Environment="KUBELET\_SYSTEM\_PODS\_ARGS=--pod-manifest-path=/etc/kubernetes/manifests --allow-privileged=true"

Environment="KUBELET\_NETWORK\_ARGS=--network-plugin=cni --cni-conf-dir=/etc/cni/net.d --cni-bin-dir=/opt/cni/bin"

Environment="KUBELET\_DNS\_ARGS=--cluster-dns=[10.96.0.10](https://l.facebook.com/l.php?u=http%3A%2F%2F10.96.0.10%2F&h=ATMIgFGOz-Zrqe2Zoj9olAqNVRLW9D-NHUMOwSGygLY1DhTLGVsZCJgCFElo3QeFgTbTKFg_SOcQWRfj5SLNcZM3KO7wL39pk0qaenluu6-DsaaeylZf) --cluster-domain=cluster.local"

Environment="KUBELET\_AUTHZ\_ARGS=--authorization-mode=Webhook --client-ca-file=/etc/kubernetes/pki/ca.crt"

Environment="KUBELET\_CADVISOR\_ARGS=--cadvisor-port=0"

Environment="KUBELET\_CERTIFICATE\_ARGS=--rotate-certificates=true --cert-dir=/var/lib/kubelet/pki"

ExecStart=

ExecStart=/usr/bin/kubelet $KUBELET\_KUBECONFIG\_ARGS $KUBELET\_SYSTEM\_PODS\_ARGS $KUBELET\_NETWORK\_ARGS $KUBELET\_DNS\_ARGS $KUBELET\_AUTHZ\_ARGS $KUBELET\_CADVISOR\_ARGS $KUBELET\_CERTIFICATE\_ARGS $KUBELET\_EXTRA\_ARGS

unable to get URL "[https://dl.k8s.io/release/stable-1.10.txt](https://l.facebook.com/l.php?u=https%3A%2F%2Fdl.k8s.io%2Frelease%2Fstable-1.10.txt&h=ATMIgFGOz-Zrqe2Zoj9olAqNVRLW9D-NHUMOwSGygLY1DhTLGVsZCJgCFElo3QeFgTbTKFg_SOcQWRfj5SLNcZM3KO7wL39pk0qaenluu6-DsaaeylZf)": Get [https://dl.k8s.io/release/stable-1.10.txt](https://l.facebook.com/l.php?u=https%3A%2F%2Fdl.k8s.io%2Frelease%2Fstable-1.10.txt&h=ATMIgFGOz-Zrqe2Zoj9olAqNVRLW9D-NHUMOwSGygLY1DhTLGVsZCJgCFElo3QeFgTbTKFg_SOcQWRfj5SLNcZM3KO7wL39pk0qaenluu6-DsaaeylZf): dial tcp: lookup dl.k8s.io on [192.168.0.1:53](https://l.facebook.com/l.php?u=http%3A%2F%2F192.168.0.1%3A53%2F&h=ATMIgFGOz-Zrqe2Zoj9olAqNVRLW9D-NHUMOwSGygLY1DhTLGVsZCJgCFElo3QeFgTbTKFg_SOcQWRfj5SLNcZM3KO7wL39pk0qaenluu6-DsaaeylZf): read udp [192.168.0.118:60959](http://192.168.0.118:60959/)->[192.168.0.1:53](http://192.168.0.1:53/): i/o timeout

Pensamos em usar minekube mas ele só funciona em arquitetura x64.

Com isso tentamos compilar o kubernetes e adaptar no orange pi seguindo estes tutoriais:

<http://www.sonic2kworld.com/blog/setting-up-go-on-armbian>

<https://gist.github.com/Adron/4e1e7d0f71da7c415f455d5930ea94c9>

<https://github.com/Project31/kubernetes-installer-rpi>

ocorreu o seguinte erro:

Step 2/16 : RUN touch /kube-build-image

---> Running in 6c39284d3e68

standard\_init\_linux.go:190: exec user process caused "exec format error"

The command '/bin/sh -c touch /kube-build-image' returned a non-zero code: 1

To retry manually, run:

docker build -t kube-build:build-b533df1e68-5-v1.10.2-1 --pull=false /root/kubernetes/\_output/images/kube-build:build-b533df1e68-5-v1.10.2-1

!!! [0606 21:25:52] Call tree:

!!! [0606 21:25:52]  1: build/release.sh:35 kube::build::build\_image(...)

Makefile:408: recipe for target 'quick-release' failed

make: \* [quick-release] Error 1

**mas não conseguimos uma solução!**

Voltamos a tentar o método de instalar o k82 na virtualbox com ubuntu mas de uma forma diferente e não obtivemos sucesso.

Lista de todos os códigos usados ao longo da implementação:

apt-get update;\  
apt-get install -y apt-transport-https ca-certificates curl software-properties-common;\  
apt-get update && apt-get install -y apt-transport-https curl ;\  
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -;\  
echo "deb http://apt.kubernetes.io/ kubernetes-xenial main"> /etc/apt/sources.list.d/kubernetes.list;\

apt-get update;\

apt-get install libltdl7;\

wget -c -O docker-ce.deb <https://download.docker.com/linux/debian/dists/jessie/pool/stable/armhf/docker-ce_17.03.0~ce-0~debian-jessie_armhf.deb> ;\

sudo dpkg -i docker-ce.deb;\  
apt-get install -y kubelet kubeadm kubectl;\

cat /etc/systemd/system/kubelet.service.d/10-kubeadm.conf;\

docker info | grep -i cgroup;\  
systemctl daemon-reload;\  
systemctl restart kubelet;\

sudo swapoff -a;\

kubeadm init

metodo 2

sudo add-apt-repository ppa:ubuntu-lxc/lxd-stable;\

sudo apt-get update;\

sudo apt-get -y upgrade;\

sudo apt-get install golang -y;\

mkdir /usr/local/go;\

export GOROOT=/usr/local/go;\

export GOPATH=$HOME;\

export PATH=$GOPATH/bin:$GOROOT/bin:$PATH;\

go get -d k8s.io/kubernetes;\

cd $GOPATH/src/k8s.io/kubernetes;\

make;\

git clone [https://github.com/kubernetes/kubernetes;\](https://github.com/kubernetes/kubernetes;%5C)

cd kubernetes;\

make quick-release;\

metodo 3

para o master use:

sudo apt-get install docker -y;\

sudo update-command-not-found;\

git clone <https://github.com/Project31/kubernetes-installer-rpi.git> ;\

cd ./kubernetes-installer-rpi ;\

bash build-master.sh

para o worker use:

sudo apt-get install docker -y;\

sudo update-command-not-found;\

git clone <https://github.com/Project31/kubernetes-installer-rpi.git> ;\

cd ./kubernetes-installer-rpi ;\

bash build-worker.sh

1- Objetivo:

Utilizar com sucesso os programas de conteinerização Kubernetes e Docker para testes de softwares usando containers.

2- Programa/versão utilizado

Putty(0.70), Etcer(1.4.4) , kubernetes(lts), docker(17.03)

3- Cenário de instalação/configuração

descrever todos os passos feitos, o mais fiel possível para que possa ser reprodutível, para chegar ao objetivo proposto

4- Resultados alcançados

Muita dificuldade para alcançar resultados até o atual momento.

5- Referência

<https://github.com/luxas/kubernetes-on-arm>

<https://stackoverflow.com/questions/40082608/how-to-delete-interface-docker0><https://kubernetes.io/docs/tasks/tools/install-minikube/><https://docs.docker.com/install/linux/docker-ce/debian/#set-up-the-repository>

<https://stackoverflow.com/questions/47018164/how-to-install-an-specific-version-of-docker-on-amazon-linux-ami>

[github.com/kubernetes-incubator/cri-tools/cmd/crictl](http://github.com/kubernetes-incubator/cri-tools/cmd/crictl)

<https://diyprojects.io/armbian-update-kernel-test-versions-nightly/#.WvnrG3Uvw3E>

<https://www.cyberciti.biz/faq/debian-ubuntu-building-installing-a-custom-linux-kernel/><https://docs.docker.com/>

<https://kubernetes.io/docs/setup/independent/install-kubeadm/#installing-kubeadm-kubelet-and-kubectl>

<https://github.com/docker/docker-ce/blob/master/components/engine/project/ARM.md>

<https://github.com/ulm0/k8s-arm>

<https://github.com/eon01/DockerCheatSheet>